A biased random-key genetic algorithm to the maximum cardinality quasi-clique problem

Bruno Q. Pinto[1], Alexandre Plastino[2], Celso C. Ribeiro[2], Isabel Rosseti[2]

[1] Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro, Uberlândia, MG 38411-104, Brazil
{bruno.queiroz}@iftm.edu.br

[2] Universidade Federal Fluminense, Institute of Computing, Niterói, RJ 24210-240, Brazil
{plastino,celso,rosseti}@ic.uff.br

Given a graph $G = (V, E)$, the maximum cardinality quasi-clique problem amounts to finding a maximum cardinality subset C^* of the nodes in V such that the density of the graph induced in G by C^* is greater than or equal to a given threshold. This problem has a number of applications in data mining, e.g., in social networks or phone call graphs. We propose a biased random-key genetic algorithm for this problem. Computational experiments show that the proposed biased random-key genetic algorithm outperforms an existing iterated greedy heuristic.

1 Introduction and notation

Let $G = (V(G), E(G))$ be a graph defined by a node set $V(G)$ and an edge set $E(G) \subseteq V(G) \times V(G)$. A graph is complete if there is an edge in $E(G)$ connecting every two different nodes in $V(G)$. A graph $H = (V(H), E(H))$ is a subgraph of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$, which is denoted by $H \subseteq G$. The graph $G(V')$ induced in G by $V' \subseteq V(G)$ is that with node set V' and edge set formed by all edges of $E(G)$ with both extremities in V'. Let $\mathcal{N}_G(v) = \{u \in V(G) : (u, v) \in E(G)\}$ be the subset of nodes adjacent to a node $v \in V(G)$. The degree of a node $v \in V(G)$ is given by $deg_G(v) = |\mathcal{N}_G(v)|$. The density of graph G is given by $dens(G) = |E(G)|/(|V(G)| \times (|V(G)| - 1)/2)$.

A subset $C \subseteq V(G)$ is a clique of G if the graph $G(C)$ induced in G by C is complete. Given a graph G and a threshold γ, the *maximum cardinality quasi-clique problem* amounts to finding a maximum cardinality subset of nodes $C^* \subseteq V(G)$ such that the density of the graph $G(C^*)$ is greater than or equal to γ. This problem is *NP*-hard, since it admits the maximum clique problem as a special case in which $\gamma = 1$ [9]. It has applications in data mining, e.g., in social networks or phone call graphs [1].

Oliveira et al. [7, 8] proposed some constructive heuristics for this problem. They start from initial solutions built by a potential-based greedy randomized heuristic, which is an adaptation of the construction phase of the algorithm in [1], and alternate between two phases: partial destruction of the current solution and reconstruction of a feasible solution using a greedy randomized adaptive algorithm to complete the part of the solution that was destroyed. Among several variants derived from the iterated greedy approach (random destruction followed by greedy reconstruction) of Ruiz and Stützle [11], the repeated application of greedy destruction followed by greedy reconstruction (RIG*) led to the best results in terms of solution quality [8].

2 Biased random-key genetic algorithm

Genetic algorithms with random keys, or random-key genetic algorithms (RKGA), were first introduced by Bean [2]. Two parents are selected at random from the entire population to implement the cross-over operation in the implementation of a RKGA. Parents are allowed to be selected for mating more than once in a given generation. A biased random-key genetic algorithm (BRKGA see [3] for a review) differs from a RKGA in the way parents are selected for crossover. In a BRKGA, each element is generated combining one element selected at random from the elite solutions in the current population, while the other is a non-elite solution. Selection is biased, since one parent is always an elite individual. Biased random-key genetic algorithms have been successfully applied to many combinatorial optimization problems.

The algorithm evolves a population of chromosomes formed by $|V(G)|$ real numbers (random keys) in the range $[0, 1]$ that are randomly generated in the initial population. The fitness of the chromosome is given by the cost of the solution found by a decoding heuristic that receives the random keys as input. The decoding heuristic is a modified version of the construction algorithm in [1], which selects the next node to be inserted into the current solution from a restricted candidate list based either on the vertex degrees or on the potential of improvement associated with each vertex.

At each new generation, the population is partitioned into two sets: TOP and $REST$. The size of the population is $|TOP| + |REST|$. The best solutions are kept in TOP while the others are placed in $REST$. The chromosomes in TOP are copied, without change, to the population of the next generation. New mutants are placed in set BOT. The remaining elements of the new population are obtained by crossover with one parent randomly chosen from TOP and the other from $REST$. Both parents are selected at random from the entire population. Since a parent solution can be chosen for crossover more than once in a given generation, elite solutions have a higher probability of passing their random keys to the next generation. In this way, $|REST| - |BOT|$ offspring solutions are created [6]. The sizes of sets TOP, $REST$, and BOT are parameters that must be tuned.

3 Numerical results

We report computational experiments comparing the results obtained by the iterated greedy heuristic RIG* [8] and the BRKGA heuristic proposed in this work. Since specific test instances for the maximum cardinality quasi-clique problem are not available in the literature, we proposed difficult instances derived from instances of the maximum clique problem [4, 10]. The two heuristics were coded in C and complied using gcc version 4.9.1. All tests were carried out on a personal computer Dell Studio i5-450M CPU 2.40GHz with 6 GB RAM running the Linux operating system Ubuntu 14.04.1 LTS. The parallel capability of the processor has not been used.

The parameters of the BRKGA heuristic have been tuned using the IRACE tool [5] and the best settings appear in Table 1. Heuristics RIG* and BRKGA were run ten times for each instances. RIG* was made to stop after 100 iterations without improvement in the best solution [7]. The average computation time observed for RIG* over the ten runs for each problem test was used as the stopping criteria of the BRKGA approach. Therefore, the results obtained by RIG* and BRKGA can be compared in terms of solution quality, since on average both heuristics run for the same computation time. Numerical results are presented in Table 2, where the best results for each instance are displayed in bold face.

Table 1: Parameter settings for the BRKGA heuristics (obtained with IRACE [5]).

Parameter description	Value
Population size	73
Fraction of the population in the TOP set	0.18
Fraction of the population replaced by mutants	0.20
Probability that an offspring inherits an allele from the elite parent	0.92

4 Concluding remarks

Table 3 summarizes the main results from the computational experiments. These results show that the biased random-key genetic algorithm BRKGA delivers the best results with respect to the four measures used to compare the two approaches. Further experiments and results will be presented in the final, extended version of this paper.

Table 2: Computational results obtained by the heuristics.

	RIG*		BRKGA	
Instance	Best	Mean	Best	Mean
sanr400_0.7	**30**	28.7	**30**	**29.3**
sanr400_0.5	31	30.0	**32**	**31.6**
sanr200_0.9	91	90.9	**92**	**91.1**
sanr200_0.7	72	72.0	**73**	**72.2**
San400_0.7_3	38	36.4	**40**	**37.2**
San400_0.7_2	**62**	**62.0**	**62**	**62.0**
San400_0.7_1	**201**	**201.0**	**201**	**201.0**
San400_0.5_1	**400**	**400.0**	**400**	**400.0**
San200_0.9_3	**37**	36.3	**37**	**36.8**
San200_0.9_2	**55**	46.2	**55**	**51.4**
San200_0.9_1	50	49.5	**54**	**52.1**
San200_0.7_2	**34**	**34.0**	**34**	**34.0**
San200_0.7_1	**57**	55.8	**57**	**57.0**
p_hat1000-1	142	141.1	**144**	**143.5**
p_hat700-1	**118**	116.4	**118**	**117.8**
p_hat500-1	95	94.6	**96**	**95.8**
p_hat300-1	63	62.3	**64**	**63.1**
MANN_a27	**135**	**134.6**	133	132.1
keller4	**54**	**52.7**	51	51.0
Gen400_0.9_65	51	49.2	**52**	**51.8**
Gen400_0.9_55	51	**50.7**	**52**	50.6
Brock800_1	87	85.1	**91**	**89.9**
Brock400_2	**185**	183.2	**185**	**184.3**
Brock400_1	**188**	186.6	**188**	**187.1**
Brock200_2	23	22.4	**24**	**23.1**
Brock200_1	**114**	113.0	**114**	**113.5**
DSJC500.5	31	30.2	**33**	**32.2**
Frb45-21-5	111	109.2	**116**	**113.8**
Frb45-21-1	114	111.4	**119**	**115.5**
Frb40-19-5	92	89.7	**96**	**95.1**
Frb40-19-1	102	100.3	**107**	**104.4**
Frb35-17-5	73	71.3	**77**	**74.9**
Frb35-17-4	74	72.8	**78**	**76.2**
Frb35-17-2	70	68.4	**73**	**70.8**
Frb35-17-1	73	70.2	**75**	**74.3**
Frb30-15-5	56	54.2	**58**	**57.2**
Frb30-15-4	55	52.8	**58**	**55.6**
Frb30-15-2	55	53.6	**57**	**55.9**
Frb30-15-1	56	54.7	**57**	**56.3**
C500.9	55	54.0	**57**	**56.2**

References

[1] J. Abello, M. Resende, and S. Sudarsky. Massive quasi-clique detection. In J. Abello and J. Vitter, editors, *Proceedings of the 5th Latin American Symposium on Theoretical Informatics*, pages 598–612. Springer, 2002.

[2] J. C. Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal*

Table 3: Comparative summary of the numerical results obtained by BRKGA and RIG*.

	RIG*	BRKGA
Number of instances for each the heuristic found the best known solution	14	38
Number of instances for each the heuristic found the best average value	7	37
Number of runs for each the heuristic found the best known solution	65	150
Average deviation from the best known solution value over all runs	4.86%	2.30%

on Computing, 2:154–160, 1994.

[3] J. F. Gonçalves and M. G. C. Resende. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17:487–525, 2011.

[4] D. S. Johnson and M. A. Trick, editors. *Second DIMACS Implementation Challenge: Cliques, Coloring and Satisfiability*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, Providence, 1996.

[5] M. López-Ibánez, J. Dubois-Lacoste, T. Stützle, and M. Birattari. The Irace package: Iterated race for automatic algorithm configuration, 2011. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium.

[6] T.F. Noronha, M. G. C. Resende, and C. C. Ribeiro. A biased random-key genetic algorithm for routing and wavelength assignment. *Journal of Global Optimization*, 50:503–518, 2011.

[7] A. B. Oliveira. Heurísticas para o problema de quasi-clique de cardinalidade máxima. Master's thesis, Universidade Federal Fluminense, Niterói, Brazil, 2013.

[8] A. B. Oliveira, A. Plastino, and C. C. Ribeiro. Construction heuristics for the maximum cardinality quasi-clique problem. In *Abstracts of the 10th Metaheuristics International Conference*, page 84, Singapore, 2013.

[9] J. Pattillo, A. Veremyev, S. Butenko, and V. Boginski. On the maximum quasi-clique problem. *Discrete Applied Mathematics*, 161:244–257, 2013.

[10] W. Pullan, F. Mascia, and M. Brunato. Cooperating local search for the maximum clique problem. *Journal of Heuristics*, 17:181–199, 2011.

[11] R. Ruiz and T. Stützle. A simple and effective iterated greedy algorithm for the permutation flow-shop scheduling problem. *European Journal of Operational Research*, 177:2033–2049, 2006.